Frank Buschmann
Alejandro P. Buchmann
Mariano A. Cilia (Eds.)

LNCS 3013

Object-Oriented Technology

ECOOP 2003 Workshop Reader

ECOOP 2003 Workshops
Darmstadt, Germany, July 2003
Final Reports

Springer

Frank Buschmann Alejandro P. Buchmann
Mariano A. Cilia (Eds.)

Object-Oriented Technology

ECOOP 2003 Workshop Reader

ECOOP 2003 Workshops
Darmstadt, Germany, July 21-25, 2003
Final Reports

Springer

Volume Editors

Frank Buschmann
Siemens Corporate Technology, Germany
Otto-Hahn-Ring 6, 81739 Munich, Germany
E-mail: Frank.Buschmann@siemens.com

Alejandro P. Buchmann
Mariano A. Cilia
Darmstadt University of Technology, Germany
FB Informatik, Databases and Distributed Systems
Hochschulstr. 10, 64289 Darmstadt, Germany
E-mail: buchmann@informatik.tu-darmstadt.de
cilia@dvs1.informatik.tu-darmstadt.de

Library of Congress Control Number: 2004108314

CR Subject Classification (1998): D.1-3, H.2, F.3, C.2, K.4, J.1

ISSN 0302-9743
ISBN 3-540-22405-X Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Protago-TeX-Production GmbH
Printed on acid-free paper SPIN: 11299011 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3013

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Preface

This volume represents the seventh edition of the ECOOP Workshop Reader, a compendium of workshop reports from the 17th European Conference on Object-Oriented Programming (ECOOP 2003), held in Darmstadt, Germany, during July 21–25, 2003.

The workshops were held during the first two days of the conference. They cover a wide range of interesting and innovative topics in object-oriented technology and offered the participants an opportunity for interaction and lively discussion. Twenty-one workshops were selected from a total of 24 submissions based on their scientific merit, the actuality of the topic, and their potential for a lively interaction. Unfortunately, one workshop had to be cancelled.

Special thanks are due to the workshop organizers who recorded and summarized the discussions. We would also like to thank all the participants for their presentations and lively contributions to the discussion: they made this volume possible. Last, but not least, we wish to express our appreciation to the members of the organizing committee who put in countless hours setting up and coordinating the workshops.

We hope that this snapshot of current object-oriented technology will prove stimulating to you.

October 2003

Frank Buschmann
Alejandro Buchmann
Mariano Cilia

Organization

ECOOP 2003 was organized by the Software Technology Group, Department of Computer Science, Darmstadt University of Technology under the auspices of AITO (Association Internationale pour les Technologies Objets) in cooperation with ACM SIGPLAN.

The proceedings of the main conference were published as LNCS 2743.

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Workshop Chairs

Frank Buschmann (Siemens Corporate Technology, Germany)
Alejandro Buchmann (Darmstadt University of Technology, Germany)

Lecture Notes in Computer Science

For information about Vols. 1–3022

please contact your bookseller or Springer-Verlag

Vol. 3125: D. Kozen (Ed.), Mathematics of Program Construction. X, 401 pages. 2004.

Vol. 3123: A. Belz, R. Evans, P. Piwek (Eds.), Natural Language Generation. X, 219 pages. 2004. (Subseries LNAI).

Vol. 3120: J. Shawe-Taylor, Y. Singer (Eds.), Learning Theory. X, 648 pages. 2004. (Subseries LNAI).

Vol. 3118: K. Miesenberger, J. Klaus, W. Zagler, D. Burger (Eds.), Computer Helping People with Special Needs. XXIII, 1191 pages. 2004.

Vol. 3116: C. Rattray, S. Maharaj, C. Shankland (Eds.), Algebraic Methodology and Software Technology. XI, 569 pages. 2004.

Vol. 3114: R. Alur, D.A. Peled (Eds.), Computer Aided Verification. XII, 536 pages. 2004.

Vol. 3113: J. Karhumäki, H. Maurer, G. Paun, G. Rozenberg (Eds.), Theory Is Forever. X, 283 pages. 2004.

Vol. 3112: H. Williams, L. MacKinnon (Eds.), New Horizons in Information Management. XII, 265 pages. 2004.

Vol. 3111: T. Hagerup, J. Katajainen (Eds.), Algorithm Theory - SWAT 2004. XI, 506 pages. 2004.

Vol. 3109: S.C. Sahinalp, S. Muthukrishnan, U. Dogrusoz (Eds.), Combinatorial Pattern Matching. XII, 486 pages. 2004.

Vol. 3107: J. Bosch, C. Krueger (Eds.), Software Reuse: Methods, Techniques and Tools. XI, 339 pages. 2004.

Vol. 3105: S. Göbel, U. Spierling, A. Hoffmann, I. Iurgel, O. Schneider, J. Dechau, A. Feix (Eds.), Technologies for Interactive Digital Storytelling and Entertainment. XVI, 304 pages. 2004.

Vol. 3104: R. Kralovic, O. Sykora (Eds.), Structural Information and Communication Complexity. X, 303 pages. 2004.

Vol. 3103: K. Deb (Ed.), Genetic and Evolutionary Computation - GECCO 2004. XLIX, 1439 pages. 2004.

Vol. 3102: K. Deb (Ed.), Genetic and Evolutionary Computation - GECCO 2004. L, 1445 pages. 2004.

Vol. 3101: M. Masoodian, S. Jones, B. Rogers (Eds.), Computer Human Interaction. XIV, 694 pages. 2004.

Vol. 3099: J. Cortadella, W. Reisig (Eds.), Applications and Theory of Petri Nets 2004. XI, 505 pages. 2004.

Vol. 3098: J. Desel, W. Reisig, G. Rozenberg (Eds.), Lectures on Concurrency and Petri Nets. VIII, 849 pages. 2004.

Vol. 3097: D. Basin, M. Rusinowitch (Eds.), Automated Reasoning. XII, 493 pages. 2004. (Subseries LNAI).

Vol. 3096: G. Melnik, H. Holz (Eds.), Advances in Learning Software Organizations. X, 173 pages. 2004.

Vol. 3094: A. Nürnberger, M. Detyniecki (Eds.), Adaptive Multimedia Retrieval. VIII, 229 pages. 2004.

Vol. 3093: S.K. Katsikas, S. Gritzalis, J. Lopez (Eds.), Public Key Infrastructure. XIII, 380 pages. 2004.

Vol. 3092: J. Eckstein, H. Baumeister (Eds.), Extreme Programming and Agile Processes in Software Engineering. XVI, 358 pages. 2004.

Vol. 3091: V. van Oostrom (Ed.), Rewriting Techniques and Applications. X, 313 pages. 2004.

Vol. 3089: M. Jakobsson, M. Yung, J. Zhou (Eds.), Applied Cryptography and Network Security. XIV, 510 pages. 2004.

Vol. 3086: M. Odersky (Ed.), ECOOP 2004 – Object-Oriented Programming. XIII, 611 pages. 2004.

Vol. 3085: S. Berardi, M. Coppo, F. Damiani (Eds.), Types for Proofs and Programs. X, 409 pages. 2004.

Vol. 3084: A. Persson, J. Stirna (Eds.), Advanced Information Systems Engineering. XIV, 596 pages. 2004.

Vol. 3083: W. Emmerich, A.L. Wolf (Eds.), Component Deployment. X, 249 pages. 2004.

Vol. 3080: J. Desel, B. Pernici, M. Weske (Eds.), Business Process Management. X, 307 pages. 2004.

Vol. 3079: Z. Mammeri, P. Lorenz (Eds.), High Speed Networks and Multimedia Communications. XVIII, 1103 pages. 2004.

Vol. 3078: S. Cotin, D.N. Metaxas (Eds.), Medical Simulation. XVI, 296 pages. 2004.

Vol. 3077: F. Roli, J. Kittler, T. Windeatt (Eds.), Multiple Classifier Systems. XII, 386 pages. 2004.

Vol. 3076: D. Buell (Ed.), Algorithmic Number Theory. XI, 451 pages. 2004.

Vol. 3074: B. Kuijpers, P. Revesz (Eds.), Constraint Databases and Applications. XII, 181 pages. 2004.

Vol. 3073: H. Chen, R. Moore, D.D. Zeng, J. Leavitt (Eds.), Intelligence and Security Informatics. XV, 536 pages. 2004.

Vol. 3072: D. Zhang, A.K. Jain (Eds.), Biometric Authentication. XVII, 800 pages. 2004.

Vol. 3071: A. Omicini, P. Petta, J. Pitt (Eds.), Engineering Societies in the Agents World. XIII, 409 pages. 2004. (Subseries LNAI).

Vol. 3070: L. Rutkowski, J. Siekmann, R. Tadeusiewicz, L.A. Zadeh (Eds.), Artificial Intelligence and Soft Computing - ICAISC 2004. XXV, 1208 pages. 2004. (Subseries LNAI).

Vol. 3068: E. André, L. Dybkj{\}ae r, W. Minker, P. Heisterkamp (Eds.), Affective Dialogue Systems. XII, 324 pages. 2004. (Subseries LNAI).

Vol. 3067: M. Dastani, J. Dix, A. El Fallah-Seghrouchni (Eds.), Programming Multi-Agent Systems. X, 221 pages. 2004. (Subseries LNAI).

Vol. 3066: S. Tsumoto, R. Słowiński, J. Komorowski, J.W. Grzymala-Busse (Eds.), Rough Sets and Current Trends in Computing. XX, 853 pages. 2004. (Subseries LNAI).

Vol. 3065: A. Lomuscio, D. Nute (Eds.), Deontic Logic in Computer Science. X, 275 pages. 2004. (Subseries LNAI).

Vol. 3064: D. Bienstock, G. Nemhauser (Eds.), Integer Programming and Combinatorial Optimization. XI, 445 pages. 2004.

Vol. 3063: A. Llamosí, A. Strohmeier (Eds.), Reliable Software Technologies - Ada-Europe 2004. XIII, 333 pages. 2004.

Vol. 3062: J.L. Pfaltz, M. Nagl, B. Böhlen (Eds.), Applications of Graph Transformations with Industrial Relevance. XV, 500 pages. 2004.

Vol. 3061: F.F. Ramas, H. Unger, V. Larios (Eds.), Advanced Distributed Systems. VIII, 285 pages. 2004.

Vol. 3060: A.Y. Tawfik, S.D. Goodwin (Eds.), Advances in Artificial Intelligence. XIII, 582 pages. 2004. (Subseries LNAI).

Vol. 3059: C.C. Ribeiro, S.L. Martins (Eds.), Experimental and Efficient Algorithms. X, 586 pages. 2004.

Vol. 3058: N. Sebe, M.S. Lew, T.S. Huang (Eds.), Computer Vision in Human-Computer Interaction. X, 233 pages. 2004.

Vol. 3057: B. Jayaraman (Ed.), Practical Aspects of Declarative Languages. VIII, 255 pages. 2004.

Vol. 3056: H. Dai, R. Srikant, C. Zhang (Eds.), Advances in Knowledge Discovery and Data Mining. XIX, 713 pages. 2004. (Subseries LNAI).

Vol. 3055: H. Christiansen, M.-S. Hacid, T. Andreasen, H.L. Larsen (Eds.), Flexible Query Answering Systems. X, 500 pages. 2004. (Subseries LNAI).

Vol. 3054: I. Crnkovic, J.A. Stafford, H.W. Schmidt, K. Wallnau (Eds.), Component-Based Software Engineering. XI, 311 pages. 2004.

Vol. 3053: C. Bussler, J. Davies, D. Fensel, R. Studer (Eds.), The Semantic Web: Research and Applications. XIII, 490 pages. 2004.

Vol. 3052: W. Zimmermann, B. Thalheim (Eds.), Abstract State Machines 2004. Advances in Theory and Practice. XII, 235 pages. 2004.

Vol. 3051: R. Berghammer, B. Möller, G. Struth (Eds.), Relational and Kleene-Algebraic Methods in Computer Science. X, 279 pages. 2004.

Vol. 3050: J. Domingo-Ferrer, V. Torra (Eds.), Privacy in Statistical Databases. IX, 367 pages. 2004.

Vol. 3049: M. Bruynooghe, K.-K. Lau (Eds.), Program Development in Computational Logic. VIII, 539 pages. 2004.

Vol. 3047: F. Oquendo, B. Warboys, R. Morrison (Eds.), Software Architecture. X, 279 pages. 2004.

Vol. 3046: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1016 pages. 2004.

Vol. 3045: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1040 pages. 2004.

Vol. 3044: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1140 pages. 2004.

Vol. 3043: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications – ICCSA 2004. LIII, 1180 pages. 2004.

Vol. 3042: N. Mitrou, K. Kontovasilis, G.N. Rouskas, I. Iliadis, L. Merakos (Eds.), NETWORKING 2004, Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications. XXXIII, 1519 pages. 2004.

Vol. 3040: R. Conejo, M. Urretavizcaya, J.-L. Pérez-de-la-Cruz (Eds.), Current Topics in Artificial Intelligence. XIV, 689 pages. 2004. (Subseries LNAI).

Vol. 3039: M. Bubak, G.D.v. Albada, P.M. Sloot, J.J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 1271 pages. 2004.

Vol. 3038: M. Bubak, G.D.v. Albada, P.M. Sloot, J.J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 1311 pages. 2004.

Vol. 3037: M. Bubak, G.D.v. Albada, P.M. Sloot, J.J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 745 pages. 2004.

Vol. 3036: M. Bubak, G.D.v. Albada, P.M. Sloot, J.J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 713 pages. 2004.

Vol. 3035: M.A. Wimmer (Ed.), Knowledge Management in Electronic Government. XII, 326 pages. 2004. (Subseries LNAI).

Vol. 3034: J. Favela, E. Menasalvas, E. Chávez (Eds.), Advances in Web Intelligence. XIII, 227 pages. 2004. (Subseries LNAI).

Vol. 3033: M. Li, X.-H. Sun, Q. Deng, J. Ni (Eds.), Grid and Cooperative Computing. XXXVIII, 1076 pages. 2004.

Vol. 3032: M. Li, X.-H. Sun, Q. Deng, J. Ni (Eds.), Grid and Cooperative Computing. XXXVII, 1112 pages. 2004.

Vol. 3031: A. Butz, A. Krüger, P. Olivier (Eds.), Smart Graphics. X, 165 pages. 2004.

Vol. 3030: P. Giorgini, B. Henderson-Sellers, M. Winikoff (Eds.), Agent-Oriented Information Systems. XIV, 207 pages. 2004. (Subseries LNAI).

Vol. 3029: B. Orchard, C. Yang, M. Ali (Eds.), Innovations in Applied Artificial Intelligence. XXI, 1272 pages. 2004. (Subseries LNAI).

Vol. 3028: D. Neuenschwander, Probabilistic and Statistical Methods in Cryptology. X, 158 pages. 2004.

Vol. 3027: C. Cachin, J. Camenisch (Eds.), Advances in Cryptology - EUROCRYPT 2004. XI, 628 pages. 2004.

Vol. 3026: C. Ramamoorthy, R. Lee, K.W. Lee (Eds.), Software Engineering Research and Applications. XV, 377 pages. 2004.

Vol. 3025: G.A. Vouros, T. Panayiotopoulos (Eds.), Methods and Applications of Artificial Intelligence. XV, 546 pages. 2004. (Subseries LNAI).

Vol. 3024: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 621 pages. 2004.

Vol. 3023: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 611 pages. 2004.

Table of Contents

Exception Handling in Object Oriented Systems: Towards Emerging Application Areas and New Programming Paradigms 1
Alexander Romanovsky, Christophe Dony, Anand Tripathi, Jørgen Lindskov Knudsen

Parallel Object-Oriented Scientific Computing Today 11
Kei Davis, Jörg Striegnitz

Communication Abstractions for Distributed Systems 17
Antoine Beugnard, Ludger Fiege, Robert Filman, Eric Jul, Salah Sadou

.NET: The Programmer's Perspective 30
Hans-Jürgen Hoffmann

Component-Oriented Programming 34
Jan Bosch, Clemens Szyperski, Wolfgang Weck

Workshop for PhD Students in Object Oriented Programming 50
Pedro J. Clemente, Miguel A. Pérez, Sergio Lujan, Hans Reiser

Formal Techniques for Java-Like Programs 62
Susan Eisenbach, Gary T. Leavens, Peter Müller, Arnd Poetzsch-Heffter, Erik Poll

Object-Oriented Reengineering 72
Serge Demeyer, Stéphane Ducasse, Kim Mens, Adrian Trifu, Rajesh Vasa, Filip Van Rysselberghe

Mobile Object Systems: Resource-Aware Computation 86
Ciarán Bryce, Crzegorz Czajkowski

Quantitative Approaches in Object-Oriented Software Engineering 92
Fernando Brito e Abreu, Mario Piattini, Geert Poels, Houari A. Sahraoui

Composition Languages 107
Markus Lumpe, Jean-Guy Schneider, Bastiaan Schönhage, Markus Bauer, Thomas Genssler

Tools and Environments for Learning Object-Oriented Concepts 119
Isabel Michiels, Jürgen Börstler, Kim B. Bruce, Alejandro Fernández

Patterns in Teaching Software Development 130
Erzsébet Angster, Joseph Bergin, Marianna Sipos

Object-Oriented Language Engineering for the Post-Java Era 143
Wolfgang De Meuter, Stéphane Ducasse, Theo D'Hondt, Ole-Lehrman Madsen

Analysis of Aspect-Oriented Software 154
Jan Hannemann, Ruzanna Chitchyan, Awais Rashid

Modeling Variability for Object-Oriented Product Lines 165
Matthias Riebisch, Detlef Streitferdt, Ilian Pashov

Object Orientation and Web Services 179
Anthony Finkelstein, Winfried Lamerdorf, Frank Leyman, Giacomo Piccinelli, Sanjiva Weerawarana

Advancing the State of the Art in Run-Time Inspection 190
Robert Filman, Katharina Mehner, Michael Haupt

Aliasing, Confinement, and Ownership in Object-Oriented Programming 197
Dave Clarke, Sophia Drossopoulou, James Noble

Author Index 209

Exception Handling in Object Oriented Systems: Towards Emerging Application Areas and New Programming Paradigms

Alexander Romanovsky[1], Christophe Dony[2], Anand Tripathi[3], and Jørgen Lindskov Knudsen[4]

[1] School of Computing Science, University of Newcastle upon Tyne, UK
alexander.romanovsky@ncl.ac.uk
[2] Université Montpellier-II and LIRMM Laboratory, France
dony@lirmm.fr
[3] Department of Computer Science, University of Minnesota, USA
tripathi@cs.umn.edu
[4] Mjølner Informatics A/S, Denmark
jlk@mjolner.dk

Abstract. Exception handling continues to be a challenging problem in object oriented system development. One reason for this is that today's software systems are getting increasingly more complex. Moreover, exception handling is needed in a wide range of emerging application areas, sometimes requiring domain-specific models for handling exceptions. Moreover, new programming paradigms such as pervasive computing, service oriented computing, grid, ambient and mobile computing, web add new dimensions to the existing challenges in this area. The integration of exception handling mechanisms in a design needs to be based on well-founded principles and formal models to deal with the complexities of such systems and to ensure robust and reliable operation. It needs to be pursued at the very start of a design with a clear understanding of the ensuing implications at all stages, ranging from design specification, implementation, operation, maintenance, and evolution. This workshop was structured around the presentation and discussion of the various research issues in this regard to develop a common understanding of the current and future directions of research in this area.

1 Summary of Objectives and Results

There are two trends in the development of modern object oriented systems: they are getting more complex and they have to cope with an increasing number of exceptional situations. The most general way of dealing with these problems is by employing exception handling techniques. Many object oriented mechanisms for handling exceptions have been proposed but there still are serious problems in applying them in practice. These are caused by

- complexity of exception code design and analysis
- not addressing exception handling at the appropriate phases of system development
- lack of methodologies supporting the proper use of exception handling
- not developing specific mechanisms suitable for particular application domains and design paradigms.

F. Buschmann et al. (Eds.): ECOOP 2003 Workshop Reader, LNCS 3013, pp. 1–10, 2004

Following the success of ECOOP 2000 workshop[1], this workshop aimed at achieving better understanding of how exceptions should be handled in object oriented (OO) systems, including all aspects of software design and use: novel linguistic mechanisms, design and programming practices, advanced formal methods, etc.

The workshop provided a forum for discussing the unique requirements for exception handling in the existing and emerging applications, including pervasive computing, ambient intelligence, the Internet, e-science, self-repairing systems, collaboration environments. We invited submissions on research in all areas of exception handling related to object oriented systems, in particular: formalisation, distributed and concurrent systems, practical experience, mobile object systems, new paradigms (e.g. object oriented workflows, transactions, multithreaded programs), design patterns and frameworks, practical languages (Java, Ada, Smalltalk, Beta), open software architectures, aspect oriented programming, fault tolerance, component-based technologies.

We encouraged participants to report their experiences of both benefits and obstacles in using exception handling, reporting, practical results in using advanced exception handling models and the best practice in applying exception handling for developing modern applications in the existing practical settings.

The workshop was attended by 18 researchers who participated in the presentation and discussion of ten position papers and one invited talk. These presentations and discussions were grouped into four thematic sessions. The first session (the invited talk and one presentation) addressed engineering systems incorporating exception handling. The focus of the second session (three presentations) was on the specific issues related to object-orientation. In the third session (three presentations) issues related to building novel exception handling mechanisms for distributed and mobile systems were discussed. The topic of the fourth session (three presentations) was exception handling and component based system development.

2 Summary of the Call-for-Papers

The call-for-papers for this workshop emphasized its broad scope and our desire to focus the workshop discussions on problems of perceived complexity of using and understanding exception handling: Why programmers and practitioners often believe that it complicates the system design and analysis? What should be done to improve the situation? Why exception handling is the last mechanism to learn and to use? What is wrong with the current practice and education?

We invited the researchers interested in this workshop to submit their position papers aiming at understanding why exception handling mechanisms proposed and available in earlier OO languages (discussed, for example, at ECOOP 1991 Workshop on Ex-

[1] A. Romanovsky, Ch. Dony, J. L. Knudsen, A. Tripathi. Exception Handling in Object Oriented Systems. In J. Malenfant, S. Moisan, A. Moreira. (Eds.) "Object-Oriented Technology. ECOOP 2000 Workshop Reader". LNCS-1964. pp. 16-31, 2000.

ception Handling and Object-Oriented Programming[2]) are not widely used now. We were interested in papers reporting practical experiences relating both benefits and obstacles in using exception handling, experience in using advanced exception handling models, and the best practices in using exception handling for developing modern applications in existing practical settings.

The original plan was to have up to 20 participants. We asked each participant to present his/her position paper, and discuss its relevance to the workshop and possible connections to work of other attendees. The members of the organizing committee reviewed all submissions. The papers accepted for the workshop sessions were posted on the workshop webpage so the participants were able to review the entire set of papers before attending the workshop.

Additional information can be found on the workshop web page:
http://www.cs.ncl.ac.uk/~alexander.romanovsky/home.formal/ehoos2003.html

The proceedings of the workshop are published as a technical report TR 03-028 by Department of Computer Science, University of Minnesota, Minneapolis, USA:
A. Romanovsky, C. Dony, J. L. Knudsen, A. Tripathi. *Proceedings of the ECOOP 2003 Workshop on Exception Handling in Object-Oriented Systems: Towards Emerging Application Areas and New Programming Paradigms. 2003.*

3 List of the Workshop Presentations

The workshop started with an invited talk delivered by William Bail (Mitre) on *Getting Control of Exception.*

After that following position papers were discussed:

1. Ricardo de Mendonça da Silva, Paulo Asterio de C. Guerra, and Cecília M. F. Rubira (U. Campinas, Brazil). *Component Integration using Composition Contracts with Exception Handling.*

2. Darrell Reimer and Harini Srinivasan (IBM Research, USA). *Analyzing Exception Usage in Large Java Applications.*

3. Peter A. Burh and Roy Krischer (U. Waterloo, Canada). *Bound Exceptions in Object Programming.*

4. Denis Caromel and Alexandre Genoud (INRIA Sophia Antipolis, France). *Non-Functional Exceptions for Distributed and Mobile Objects.*

5. Tom Anderson, Mei Feng, Steve Riddle, and Alexander Romanovsky (U. Newcastle, UK). *Error Recovery for a Boiler System with OTS PID Controller.*

6. Joseph R. Kiniry (U. Nijmegen, Netherlands). *Exceptions in Java and Eiffel: Two Extremes in Exception Design and Application.*

---

[2] Dony, Ch., Purchase, J., Winder. R.: Exception Handling in Object-Oriented Systems. Report on ECOOP '91 Workshop W4. OOPS Messenger **3**, 2 (1992) 17-30

7. Giovanna Di Marzo Serugendo (U. Geneva, Switzerland) and Alexander Romanovsky (U. Newcastle, UK). *Using Exception Handling for Fault-Tolerance in Mobile Coordination-Based Environments.*

8. Frederic Souchon, Christelle Urtado, Sylvain Vauttier (LGI2P Nimes, France), and Christoophe Dony (LIRMM Montpellier, France). *Exception Handling in Component-based Systems: a First Study.*

9. Johannes Siedersleben (SD&M Research, Germany). *Errors and Exceptions – Rights and Responsibilities.*

10. Robert Miller and Anand Tripathi (U. Minnesota, USA). *Primitives and Mechanisms in the Guardian Model for Exception Handling in Distributed Systems.*

4 Summary of Presentations

The first sessions focused on software engineering issues in exception handling. It included two talks. William Bail (Mitre) presented the invited lecture titled *"Getting control of exceptions"*. In his talk he noted that the past developments in this field have allowed programmers to define and use exceptions and this has led a significant advantage in being able to write more reliable software. While not explicitly helping us avoid errors, they enable us to detect their presence and control their effects. Yet they act in opposition to much of what we have learned is good software design - simple structures with well-defined control flows. In addition, they complicate the process of performing formal analyses of systems. This talk elaborated on this issue and projected some potential ideas to help reconcile these challenges, especially with the use of OO concepts. The second talk in the first session was given by Johannes Siedersleben (SD&M Research, Germany). He presented the paper *"Errors and Exceptions - Rights and Responsibilities"*. The talk emphasized the strict separation of errors to be handled by the application and the 'true' exceptions which require recovering and restart mechanisms. It suggested the use of the term "emergency" for the exceptions of the second type because in many programming languages, exceptions can and are used for many non-exceptional situations. The paper also describes a component-based strategy to handle emergencies using so called safety facades.

The theme of the second session centered on exception handling in OO Systems. It included three papers. The first talk in this session was by Harini Srinivasan (IBM Research, USA), who presented the paper "Analyzing Exception Usage in Large Java Applications". This talk emphasized that proper exception usage is necessary to minimize time from problem appearance to problem isolation and diagnosis. It discusses some common trends in the use of exceptions in large Java applications that make servicing and maintaining these long running applications extremely tedious. The talk also proposes some solutions to avoid or correct these misuses of exceptions. The second presentation was by Roy Krischer (U. Waterloo, Canada) on the paper entitled *"Bound Exceptions in Object Programming"*. Many modern object-oriented languages do not incorporate exception handling within the object execution environment. Specifically, no provision is made to involve the object raising an exception in the catching mechanism in order to allow discrimination among multiple objects raising the same exception. The notion of bound exceptions is introduced, which as-

sociates a 'responsible' object with an exception during propagation and allows the catch clause to match on both the responsible object and exception. Multiple strategies for determining the responsible object were discussed in this talk, along with extending bound exceptions to resumption and non-local propagation among coroutines/tasks. The third speaker in this session was Joseph R. Kiniry (U. Nijmegen, Netherlands) who presented his paper *"Exceptions in Java and Eiffel: Two Extremes in Exception Design and Application"*. His focus was on an analysing the exception handling mechanisms in the Java and Eiffel languages and on contrasting the style and the semantics of exceptions in these two languages. The talk showed how the exception semantics impacts programming (technically) and programmers (socially). According to the author the primary result of this analysis is that Java checked exceptions are technically adequately designed but are socially a complete failure. This position is supported by an analysis of hundreds of thousands of lines of Java and Eiffel code.

The third session had three talks on exception handling in mobile and distributed systems. Alexandre Genoud (INRIA Sophia Antipolis, France) presented the paper *"Non-Functional Exceptions for Distributed and Mobile Objects"*. He proposed the notion of non-functional exceptions to signal failures occuring in non functional properties (distribution, transaction, security, etc.). He described a hierarchical model based on mobile exception handlers. Such handlers, attached to distribution-specific entities (proxies, futures), are used to create middleware-oriented handling strategies. The handling of exceptions can indifferently be at non-functional or application-level. The second speaker in this session was Alexander Romanovsky (U. Newcastle upon Tyne, UK), who presented the paper *"Using Exception Handling for Fault-Tolerance in Mobile Coordination-Based Environments"*. Mobile agent-based applications very often run on a mobile coordination-based environment, where programs communicate asynchronously through a shared memory space. The aim of this paper is to propose an exception handling model suitable for such environments. It is our view that it is conceptually wrong to treat such exceptions as usual events or tuples. This is why in the model proposed a local handler agent is created each time when an exception is signalled: this guarantees handling, allows exceptions and handlers to be dynamically associated and decreases the overall overheads. The third talk in this session was presented by Anand Tripathi (University of Minnesota, Minneapolis, USA) on *"Primitives and Mechanisms in the Guardian Model for Exception Handling in Distributed Systems."* In this talk he elaborated on notion of the guardian for encapsulating exception handling policies in a distributed application. He presented the core set of primitives of the guardian model which allow the programmer to specify and control the recovery actions of cooperating processes so that each process performs the required exception handling functions in the right context. This talk elaborated on how the various other existing models for distributed exception handling can be implemented using the guardian model.

The fourth session in the workshop focused on exception handling issues related to software and systems composition. Paulo Asterio de C. Guerra (U. Campinas, Brazil), presented the paper *"Component Integration using Composition Contracts with Exception Handling"*. He outlined an architectural solution for the development of dependable software systems out of concurrent autonomous component-systems. The solution is based on the concepts of coordination contracts and Coordinated Atomic (CA) Actions, which are adapted to a service-oriented approach. The second talk in

this session was by Mei Feng (U. Newcastle upon Tyne, UK) on the paper *"Error Recovery for a Boiler System with OTS PID Controller"*. The talk presented the protective wrapper development for the model of the system in such a way that they allow detection and tolerance of typical errors caused by unavailability of signals, violations of range limitations, and oscillations. In the presentation the case study demonstrated how forward error recovery based on exception handling can be systematically incorporated at the level of the protective wrappers. The last talk of this session was given by Christelle Urtadeo (LGI2P – Ecole des Mines d'Ales, Nimes, France) on *"Exception Handling in Component Based Systems: A First Study.* Christelle Urtado presented a preliminary study on exception handling in component-based systems written by F. Souchon, C. Urtado, S. Vauttier and C. Dony. The talk focused on the category of components that interact in a contract-based manner and communicate asynchronously. According to the authors, exception handling for such components should provide four features: handler contextualization, concurrent activity coordination, exception concertation and exception handling support for broadcasted requests. These requirements have already shown to be pertinent in a similar context: the SaGE exception handling system that has been designed by the authors for multi-agent systems. The speaker has used SaGE implementation to exemplify how exception handling should be managed for the considered components.

5 Summary of Discussions

The workshop has gathered a rich collection of contributions covering many of the subjects that constitute the exception handling and fault tolerance domains. Presentations had generated numerous questions and exchanges. The main goal of the concluding 45 minutes general discussion was to bring to the fore the main conclusions, results, challenging issues and research directions implicitly or explicitly expressed during papers presentations and discussions on the four main subjects addressed during the workshop:

- Software engineering issues in exception handling,
- Exception handling issues in today's standard object-oriented systems
- Reliable mobile, distributed, concurrent systems
- Reliable component-based systems.

Discussion on the first issue led the attendees to a primary and major conclusion that after almost thirty years of research and many years of experimentation with many different languages, it appears that our knowledge of language primitives for exception handling is somehow high but that there is a huge need for standard definitions and for standard analysis, design and programming patterns. Indeed, the primitive for exception handling in today's languages are, in one way or another, evolutions of those proposed in the seminal paper by John Goodenough written in1975. They are primitives for signaling, catching, and handling of exception based different execution models such as entailing termination (or retry), resignaling (or propagation), or resumption. As far as these crucial concepts have been correctly extracted from the foundational research, important progress have been made in their understanding, adaptation, development and implementation. Future research efforts will need to

adapt these primitives to tomorrow's needs. Unfortunately, there has never been a basic agreement, a norm, on the definition of the terms "exception", "exception handling", "fault tolerance".

Early terms such as "domain", "range" or "monitoring" exceptions as proposed in Goodenough's paper are not standardised mainly because they do not reflect the concept complexity. Researchers thus have to choose or to re-invent ever again their own definitions : consider the following terms, all coming from previous works, which have been used in our workshop contributions to denote either the same thing or subtly different things : exception, error, warning, condition, alarm, emergency, etc. There is neither a general agreement on the standard patterns to handle exceptions or to write fault-tolerant or defensive programs. When architectural solutions exist, they are not known of today's developers because there is no reference book where such patterns are described. It is interesting to note that this issue was quoted as an open issue in our ECOOP 2000 workshop conclusion, and that no significant advance has been made in that direction. However, this year, new interesting papers have reported experiences on how developers, either experienced or not, deal with exceptions, how they sometimes reinvent known solutions and make known mistakes, how they sometimes misuse or misunderstand language constructs. Everybody has thus agreed to stress the need for dedicated design patterns and to present this as a major issue.

The second main point in the discussion concerned the use of exception handling techniques in today's object-oriented systems in general and more particularly in Java as far as this language has been at the heart of the debates. Java is today "de facto" a vast field of experimentation for exception handling because it makes it mandatory for programmers to deal with exceptions, especially for the so-called "checked exceptions" (see section 4). Dealing with exception at a large scale, as experienced for example by Ada developers, had never been done by object-oriented developers because no language before Java mandated it. As William Bail noticed in the workshop introductory talk, the history of exceptions is a love/hate relationship and programmers generally do not like to handle exception for various good or wrong reasons: it makes programs longer to write, breaks code harmony, is boring, seems useless, or seems to slow execution time down.

The discussion thus focused on issues connected to Java design choices and constraints (e.g. static typing) related to exception handling. Some of these issues are already well known and sometimes related solutions exists and have been published years ago; they have however been considered here in the light of new experience reports. A first one is that handling (putting the system back into a coherent state within catch clauses body) cannot be performed in a generic way by sending messages to the exception object. Examples have been presented showing how it can impose to write several catch clauses where one would be enough instead. Besides, the exception objects do not contain enough information.

More generally, and William Bail also indirectly quoted this in his introductory talk, Java's exception handling system is certainly and for many reasons not enough object-oriented. A connected problem is that a *try block* with an empty *catch clauses* (entailing termination) can be understood by beginners, or used by developers in a